SICORE : SYSTÈME GÉNÉRAL DE CHIFFREMENT AUTOMATIQUE

Pascal Rivière

Sicore est un projet qui a été initié au printemps 1993 pour répondre à un besoin : doter l'Insee d'un outil général de chiffrement automatique, d'usage simple, tout en créant autour de cet outil l'organisation et la méthodologie nécessaires à une utilisation efficace de celui-ci.

Deux ans et demi après, ce système, en grande partie achevé, existe et a montré l'efficacité de son fonctionnement.

Présenter Sicore demeure cependant un exercice périlleux, car il multiplie les paradoxes : il désigne un logiciel, mais aussi un système ; il est très facile de l'utiliser pour coder automatiquement, mais l'outil Sicore, destiné aux experts, est d'usage très complexe ; d'ailleurs, la véritable vocation de cet outil n'est pas le chiffrement automatique, mais la mise au point des connaissances ; enfin, Sicore emploie, au moins en partie, l'algorithme Quid, mais diffère totalement du logiciel Quid et n'en réutilise aucun programme.

Pour décrire l'ensemble de l'architecture Sicore, on commencera par définir ce que l'on entend par "coder", puisqu'il s'agit tout de même de l'objectif premier. On insistera plus particulièrement sur le codage automatique, et sur la méthode Quid. La deuxième partie s'attachera à présenter le fonctionnement de Sicore en tant qu'outil de chiffrement : séparation entre programmes et connaissances, contenu des bases de connaissances, algorithmes d'apprentissage et de codage. Cependant, comme nous l'avons vu, il ne faut pas se limiter aux seuls programmes : le "système Sicore" (réutilisation des expériences de codage, mise au point des connaissances, organisation autour de Sicore) sera décrit dans la partie 3. Enfin, le dernier chapitre sera plus concret, et répondra à des questions du type : comment utilise-t-on Sicore en pratique ? Comment mettre en place le chiffrement automatique dans une enquête ? Quelles ont été les utilisations de Sicore, et avec quels résultats ?

1. Coder

1.1. Définitions

En première approximation, *chiffrer*, ou *coder*, consiste simplement à interpréter un libellé en fonction d'une nomenclature prédéfinie, en lui affectant un code de cette nomenclature. Le codage permet donc de placer dans une catégorie connue une réponse libre à un questionnaire, ce qui nous ramène à la structure d'une réponse fermée, traitable statistiquement.

D'un point de vue théorique, la classification est supposée existante (c'est la nomenclature), et le codage est donc une opération de *discrimination*.

Le codage, en pratique, peut être effectué par :

- *l'enquêté* : lorsqu'il répond à une question fermée, il effectue dans sa tête son propre codage ;

- *l'enquêteur* : il arrive qu'il ait une "carte codes", c'est-à-dire une sorte de nomenclature sur papier, qui lui permet de coder ce que lui dit l'enquêté ;

- *le codeur* : pour des variables complexes comme la profession ou l'activité, le chiffrement est effectué par des professionnels, qui codent des centaines de libellés et acquièrent donc une véritable expérience du codage ;

- *le statisticien* : dans quelques très rares cas, les codeurs n'arrivent pas à se mettre d'accord sur le chiffrement d'intitulés difficiles, et peuvent demander l'arbitrage d'un responsable de l'enquête ;

- *l'ordinateur* : chiffrement automatique.

Il importe de distinguer les concepts de *chiffrement automatique* et de *chiffrement assisté*.

Par **chiffrement automatique**, on entend algorithme de chiffrement ne nécessitant pas d'intervention humaine, et en particulier applicable en traitement par lots ; un tel algorithme, lorsqu'il réussit à chiffrer, doit renvoyer un seul code.

Le **chiffrement assisté**, quant à lui, comprend toutes les techniques facilitant la tâche de codage, réalisée au cas par cas par un professionnel appelé "codeur", ou "chiffreur". En particulier, il peut renvoyer une liste d'échos parmi lesquels le codeur choisira : on a donc plus de chances de trouver le code qu'en mode automatique, où l'on doit faire

un choix. Le chiffrement assisté peut incorporer des programmes de chiffrement automatique, appliqués de façon interactive.

Sicore, en particulier, est un **logiciel destiné au chiffrement automatique**, qui n'a nullement vocation à faire du chiffrement assisté.

1.2. Le fonctionnement du codage

La vague définition du chiffrement proposée ici appelle plusieurs questions :
D'où viennent les libellés, quelle est leur forme ?
D'où viennent les codes, quelle est leur structure ?
Comment connaît-on la correspondance entre libellés et codes ?
Utilise-t-on d'autres connaissances que le libellé pour coder ?

1.2.1. Les libellés

En statistique, la présence de libellés n'est nécessaire que si le nombre de cas (i.e. de codes) possibles est trop important pour que l'on puisse utiliser une question fermée, autrement dit lorsque la nomenclature associée est riche. Par exemple, il n'est pas utile d'exiger un libellé lorsque la question porte sur la profession en 8 postes.

Lorsque la collecte est effectuée par un enquêteur, au domicile de l'enquêté, on peut éviter les libellés lorsque la nomenclature est suffisamment simple pour être prise en compte par l'enquêteur lui-même. Dans ce cas, il dispose d'une *carte de codes* de 2 pages au maximum, décrivant la nomenclature (liste de correspondances entre libellés et codes), grâce à laquelle il chiffre lui-même. Ainsi, dans certaines enquêtes auprès des ménages, les diplômes sont codés à l'aide d'une carte codes.

Néanmoins, de nombreuses variables, comme la profession (nomenclature détaillée), l'activité économique, les causes de décès, demeurent *trop complexes* pour qu'une carte de codes suffise. Les valeurs de ces variables à l'issue de la collecte seront donc nécessairement des libellés.

Lors d'une enquête auprès des ménages, ceux-ci peuvent être renseignés par l'enquêteur (sur papier, ou directement en cas de saisie portable), ou bien par l'enquêté lui-même. Lorsque les données proviennent de sources administratives, ce sont souvent les entreprises qui les remplissent ; il arrive également que ce soient les ménages (cf. l'état civil).

Ces textes sont en général *écrits en langage naturel*, dans la langue du pays considéré. Mais il peut aussi arriver que des codes soient directement inscrits, sans que les

directives de l'enquête ne l'exigent. La difficulté provient alors du fait que ces codes sont parfois des codes internes à une entreprise ou à un établissement, ce qui rend le codage paradoxalement plus difficile.

Enfin, ces libellés sont parfois soumis à certaines restrictions, par exemple sur le nombre de caractères ou le nombre de mots.

1.2.2. Les codes

L'ensemble des codes, ou catégories, est le cadre de référence du statisticien. Pour certaines variables, cet ensemble forme, en première approximation ce que l'on appelle une **nomenclature**. Il en existe de très nombreuses : professions, activités, formations, brevets, produits, maladies, etc.

Une nomenclature est associée à un certain concept, et donc à un *champ sémantique particulier*. Les catégories statistiques de la nomenclature doivent partitionner ce champ, c'est-à-dire que deux catégories quelconques doivent avoir une intersection vide, et que toutes les catégories doivent couvrir l'ensemble du champ considéré.

Une nomenclature est gérée par un *nomenclateur*, qui peut *proposer* d'ajouter ou d'enlever telle ou telle catégorie, de modifier des dépendances entre catégories. Ces modifications ne se font pas si facilement que cela, et nécessitent la plupart du temps une validation légale, par exemple un décret.

Le responsable de la nomenclature doit également se soucier de l'évolution de cet objet complexe, du maintien de sa cohérence, ainsi que de ses correspondances avec d'autres nomenclatures du même champ.

Enfin, une nomenclature est par essence même *structurée* sous la forme d'un arbre (plus précisément d'un graphe acyclique direct), c'est-à-dire qu'elle est composée de *postes*, organisés de façon hiérarchique : il y a dans une nomenclature différents niveaux. Les postes auront en général des noms spécifiques selon ces niveaux (section, division, classe, ...). Dire d'une nomenclature qu'elle n'est qu'un ensemble de codes constitue donc une simplification, puisque la notion d'ensemble ne fait pas apparaître ces relations entre catégories, et *aplatit* par conséquent la structure.

Cette distinction de niveaux est essentielle pour le codage : en effet, lorsque l'information fournie dans le libellé est imprécise (par exemple si l'on déclare comme profession "ENSEIGNANT"), elle n'en demeure pas moins possible à coder. Simplement, le code ne pourra être complet : on ne pourra pas se situer au dernier niveau de l'arborescence, qui correspond aux codes les plus précis possibles.

Il existe également des variables susceptibles d'avoir de nombreux codes, mais où l'ensemble de ces derniers ne constitue nullement une nomenclature. C'est le cas par exemple des communes.

Des variables comme les activités ou les produits sont définies, classées, structurées, par des nomenclaturistes qui sont des spécialistes du champ sémantique de la variable. À l'inverse, la catégorisation en communes préexiste et s'impose au statisticien, qui ne peut qu'enregistrer les évolutions ; la différence vient du fait qu'*une commune n'est pas à proprement parler un concept statistique* : il s'agit déjà d'une catégorie.

En revanche, cela n'empêche pas qu'elle soit structurée, en l'occurrence en région-département-commune.

La distinction entre nomenclatures et non-nomenclatures peut paraître très éloignée du codage : ce n'est pas tout à fait exact, car les ensembles de codes, organisés hiérarchiquement ou non, représentent la *référence statistique*, le cadre de travail du chiffrement.

Savoir qui les gère et de quelle façons ces ensembles évoluent n'est donc pas indifférent.

1.2.3. Correspondance entre libellés et codes

Associer un code à un libellé nécessite de connaître la passerelle qui relie les deux.

En général, il semble qu'un tel pont existe, qui n'est autre que la nomenclature : à tout poste, à tout code, elle associe un ou plusieurs libellés possibles, et éventuellement quelques commentaires.

La pratique montre cependant qu'une telle référence ne suffit pas. En effet, les individus s'expriment dans leur propre langage, qui n'utilise pas nécessairement le vocabulaire officiel, mis au point par des experts. Le codage doit donc s'adapter au matériau dont il dispose : les réponses observées en pratique.

Il s'agit donc, pour quelqu'un qui va coder à la main, de traduire les réponses des enquêtés afin de les réinterpréter en regard de la nomenclature. Si l'on veut coder automatiquement, il faudra partir d'un ensemble important de libellés provenant de l'enquête ou de la source administrative considérée, eux-mêmes codés manuellement. Le tout constituera un fichier de référence pour le chiffrement automatique.

1.2.4. Les connaissances autres que le libellé

Pour de nombreuses variables, le libellé seul ne suffit pas toujours pour coder. Il est à la rigueur suffisant pour le codage des produits ou des activités. Mais il ne l'est même

pas pour les communes, par exemple, car la connaissance du département est parfois nécessaire pour distinguer des villages ayant le même nom.

Le cas le plus frappant est celui de la profession, où de nombreuses variables supplémentaires sont utiles : activité de l'établissement, qualification, statut, fonction, etc. Pour le codage des causes de décès, on aura également besoin d'autres informations, comme l'âge de l'individu, par exemple.

Par conséquent, le cadre général du codage n'est pas la transformation d'un libellé en un code, mais celle d'un libellé *et de valeurs de variables annexes* en un code.

Ces variables supplémentaires permettent donc de lever l'ambiguïté sur le code. L'ensemble de ces variables doit être fixé, connu à l'avance, de même que l'ensemble des valeurs possibles de chacune d'elles. En entrée du chiffrement, il est préférable qu'elles soient déjà elles-mêmes codées.

Enfin, il faut qu'elles aient la possibilité de prendre une valeur particulière : la valeur "manquant".

1.3. Le codage automatique

Chiffrer automatiquement, c'est faire réaliser le codage par une machine. Avant toute autre considération, il faut souligner que **le choix du codage automatique, au sein du processus de traitement d'une enquête, n'est pas innocent : il influe profondément sur l'organisation**.

En premier lieu, **le chiffrement automatique implique la saisie**. Or, souvent, le chiffrement est fait à la main, directement sur le dossier d'enquête. Introduire le codage automatique induit donc un surcoût, celui de la saisie de tous les libellés.

Mais ce n'est pas tout : l'efficacité du chiffrement automatique n'est jamais de 100 %. **Il faut donc prévoir une chaîne de traitement des non-codés**, c'est-à-dire se doter de programmes de codage assisté : le minimum serait une grille de saisie, dans laquelle on trouverait le libellé à coder ainsi que les variables annexes nécessaires, et un emplacement pour le code.

1.3.1. Principe des programmes de codage automatique

Un algorithme de codage automatique possède une structure très simple : en entrée, il reçoit un libellé (plus d'éventuelles variables annexes) ; en sortie, il renvoie un *écho de codage* (indiquant si le codage a réussi ou échoué, et précisant les divers types de

réussite ou d'échec), un *résultat de codage* (un code, ou pas de code, ou plusieurs codes), plus éventuellement d'autres informations que l'utilisateur voudrait conserver.

Il se décompose en général en deux étapes majeures : la **normalisation** du libellé, et la **reconnaissance** de celui-ci (au sens de la reconnaissance des formes), qui est le codage proprement dit.

La **normalisation** est un pré-traitement qui permet d'obtenir un libellé "propre", plus facile à appréhender.

Elle utilise divers types de programmes : synonymisation, élimination de mots vides, traitement de préfixes et de suffixes, phonétisation, traitement de fautes d'orthographe,...

Le logiciel de chiffrement automatique ACTR, créé par Statistique Canada, est particulièrement sophistiqué au niveau de cette partie "normalisation".

Les programmes de **reconnaissance de libellé**, quant à eux, utilisent en général la nomenclature (ou toute correspondance entre libellés et codes) en tant que donnée. Le cas le plus simple est celui où le programme ne code que les libellés qui figurent tels quels dans la nomenclature : c'est ce que l'on appelle la méthode grossière. Pour le reste, toute la difficulté est de savoir jusqu'à quel point on peut admettre un intitulé "proche" d'un intitulé de la nomenclature.

Une technique possible consiste à coder le libellé en fonction de la présence de tel ou tel *mot-clé* : en plus de la nomenclature, on se donne une liste de mots-clés, chacun étant associé à un code. Dès que le programme trouve l'un de ces mots-clés, il code le libellé complet, quels que soient les autres mots qui le constituent.

On peut aussi se définir des *mesures de distance entre textes* : dès lors qu'un libellé se trouve à une distance suffisamment faible d'un libellé de référence, on le code. Ce sont des méthodes dites de score. La mesure de distance peut être définie, par exemple, par le nombre de caractères distincts entre les deux libellés. Dans cette approche, toute la difficulté est de bien définir les seuils : à partir de quelle distance considère-t-on le libellé comme acceptable ? Ces seuils sont fonction de la variable à coder, et nécessitent une mise au point minutieuse. À l'Insee, le logiciel MCA, qui permet de déterminer des SIRET à partir de raisons sociales et adresses, utilise ce type de technique.

Très souvent, les programmes de codage automatique se déroulent en plusieurs phases : on commence par utiliser la méthode grossière, puis la méthode des mots-clés, puis une méthode de score, et enfin la méthode des mots-clés, avec un autre fichier de mots-clés. Grâce à cela, on peut associer au résultat un indicateur de qualité (plus le codage a lieu tôt dans ce processus, meilleur est l'indicateur). Ce fonctionnement en plusieurs étapes a été appliqué par les statisticiens croates, en particulier pour leur dernier recensement.

1.3.2. La méthode Quid

Quid, mis au point en 1979 par Jacques Lorigny, est une méthode générale de chiffrement, au sens où il ne dépend pas de la variable à coder. Pour cela, il fonctionne en deux temps : apprentissage et codage.

1.3.2.1. L'apprentissage

L'apprentissage s'applique à un fichier (le fichier d'apprentissage) dans lequel on associe un code à chaque libellé de la variable considérée. Il consiste à fabriquer, à partir de ce fichier, une arborescence appelée **arbre de questionnement**, qui va faciliter considérablement la deuxième étape, le codage.

Supposons que nous voulions coder automatiquement des libellés de commune. Pour cela, il existe un fichier d'apprentissage, par exemple celui des DADS. Voici quelques lignes de ce fichier :

Un fichier d'apprentissage est un fichier plat, sur chaque ligne duquel se trouvent :

Figure n° 1 :
Extrait d'un fichier d'apprentissage (FA)

```
59230FERRIERE     GDE
59230FERRIERE     GR
59230FERRIERE     GRAND
59230FERRIERE     GRANDE
59230FERRIERE     GRDE
59230FERRIERE     LAGRANDE
59231FERRIERE     PETITE
59231FERRIERE     PTE
59232FLAMENGRIE
59233FLAUMONT     WAUDRECHIES
59234FLERS        ESCREBIEU
59234FLERS        ESCREBIEUX
59236FLESQUIERES
59237FLETRE
59238FLINES       LEZ           MORTAGNE
59238FLINES       MORTAGNE
59239FLINES       LEZ           RACHES
59239FLINES       RA
59239FLINES       RACHES
59240FLOURSIES
59241FLOYON
59242FONTAINE     AU            BOIS
59243FONTAINE     AU            PIRE
59244FONTAINE     NOTRE         DAME
59246FOREST       CAMBRESIS
```

- *un libellé structuré* : un nombre fixé de mots, chaque mot ayant une longueur bien déterminée (exemple : 5 mots de 10 caractères) et commençant à une position donnée. Dans la longueur d'un mot, on tient compte des blancs ; ces mots sont en général consécutifs, mais ce n'est pas nécessaire.

- *un code* : le code associé à ce libellé

Le code et le libellé ne sont pas nécessairement disjoints, comme le montre le cas qui précède, où le libellé est formé d'un mot de 2 caractères (le département) et de 4 mots de 12 caractères (les mots du libellé de commune).

Ainsi, "59 FONTAINE NOTRE DAME 59244" est constitué des mots "59", "FONTAINE ", "NOTRE ", "DAME " et " ". Le code est ici "59244".

Pour générer un "arbre de questionnement", il faut maintenant décomposer tous les libellés du fichier en groupes de deux lettres, appelés *bigrammes*.

Voici trois exemples, dans le cas de la commune :

Puis l'on recherche, dans l'ensemble du fichier d'apprentissage, le *bigramme*[1] qui

Figure 2 :

Libellés structures

59	FO	NT	AI	NE			NO	TR	E				DA	ME									
59	FE	RR	IE	RE			GR	AN	DE														
59	FL	AU	MO	NT			WA	UD	RE	CH	IE	S											

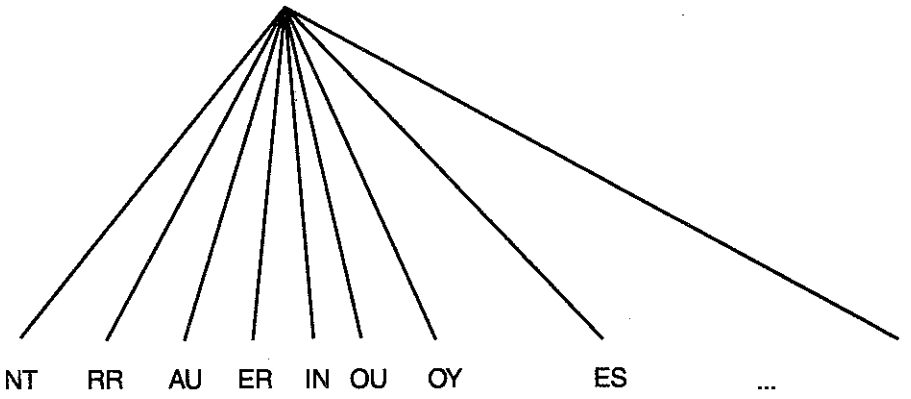
fournit le plus d'information en vue du codage ; pour cela, on calcule le gain d'information[2] apporté par chaque position de bigramme, et on prend la meilleure.

Dans la pratique, et quels que soient les libellés, on observe que c'est souvent le *deuxième bigramme du premier mot* qui apporte le plus d'information. Ceci nous donne le début de l'arborescence :

1. On devrait dire : la position de bigramme.
2. C'est-à-dire la variation d'entropie (conditionnelle), au sens de la théorie de l'information de Shannon.

Figure n°3 :

Premier niveau de l'arbre de questionnement



On n'a indiqué ici que les valeurs de deuxième bigramme observées dans les libellés de commune cités précédemment ; bien entendu, il y a énormément de valeurs possibles, donc énormément de branches possibles partant de la racine.

La branche "RR" par exemple, oriente vers le sous-fichier des libellés de communes qui ont "RR" comme deuxième bigramme du premier mot[1].

La technique consistant à rechercher le bigramme le plus informant est ensuite appliquée à chaque nœud de l'arbre, qui représente un sous-fichier du fichier initial. Par exemple, la branche de gauche, correspondant à la valeur "NT" du deuxième bigramme, pointe sur le sous-fichier de tous les libellés ayant "NT" en deuxième bigramme ; on y retrouvera "50 MONT ST MICHEL", "30 PONT ST ESPRIT", ... Dans ce sous-fichier, le bigramme apportant le plus d'information ne pourra pas être le deuxième, puisque par construction il a toujours la même valeur.

Dès lors, à chacun des noeuds de l'arbre, la "meilleure position de bigramme" sera déterminée, mais elle pourra varier d'un nœud à l'autre ; pour la branche "NT", ce sera le bigramme de département, pour la branche "RR", ce sera le premier bigramme du deuxième mot, et pour la branche "AU", le premier bigramme du premier mot.

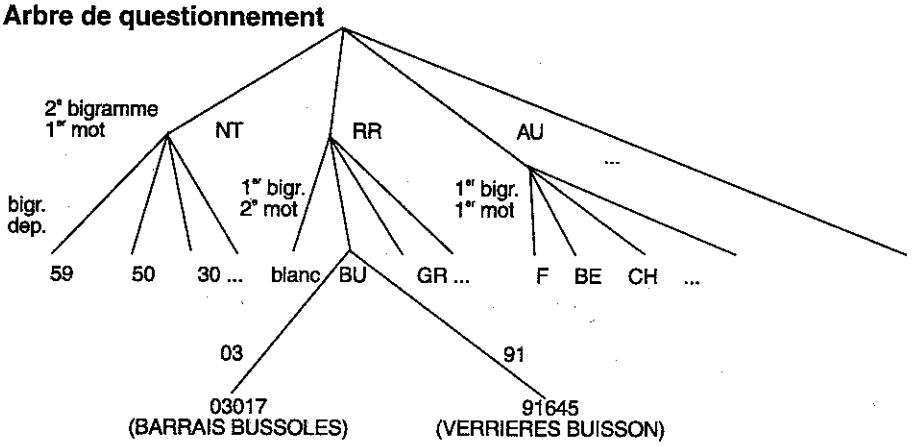
Par conséquent, le critère de choix de bigramme est un *critère local*, fonction du contexte (les bigrammes choisis jusque-là), et il s'agit là d'un des traits essentiels de la méthode Quid. En raison de cette caractéristique, elle est résolument différente des méthodes de score, où le critère est global et ne tient donc pas compte du contexte.

1. On entend par là : premier mot du libellé de commune proprement dit, i.e. hors département. Par convention, le bigramme de département sera ici le bigramme n°0.

À l'issue de l'apprentissage, on obtient un arbre[1] du type suivant :

Figure n°4 :

Arbre de questionnement



Dans le cas particulier de la branche ayant "RR" en 2e bigramme du 1er mot, et "BU" en premier bigramme du 2e mot, le schéma ci-dessus décrit la suite de la construction de l'arbre ; ainsi, il n'y a que deux communes en France ayant ces deux caractéristiques, Barrais Bussolles, dans l'Allier, et Verrières-le-Buisson, dans l'Essonne (VERRIERES BUISSON, après normalisation) ; le bigramme de département permet de discriminer entre les deux : à cet égard, il est tout aussi informant que le premier bigramme du premier mot, ou le troisième.

L'arbre se construira donc de la façon suivante : si, à un nœud donné, le sous-fichier correspondant est constitué de libellés ayant toujours le même code, on arrêtera là ; sinon, on recherchera le bigramme le plus informant, et on développera l'arbre en fonction de ce bigramme.

Dans le cas de Verrières-le-Buisson, la partie du fichier d'apprentissage formée des libellés de l'Essonne ayant "RR" en 2e bigramme du 1er mot et "BU" en 1er bigramme du 2e contient plusieurs libellés normalisés distincts : "VERRIERES BUISSON", "VERR BUISSON" "VERRIERES BUISS", "VERRIERE BUISSON" ; ils correspondent aux différents libellés que l'on peut trouver en pratique, mais sont toujours associés au code 91645. Il n'est donc nullement nécessaire de chercher à les distinguer, et il est bien légitime de stopper là le développement de l'arbre.

1. Les arbres de questionnement étant gigantesques, on ne peut, bien entendu, en présenter qu'une partie. L'extrait d'arbre présenté ne constitue qu'un cas d'école, et ne prétend pas refléter la réalité.

Prenons un autre exemple, qui nous permet de prendre conscience de **l'importance du critère local de choix de bigramme**. Les communes de Flines-lez-Mortagne et de Flines-lez-Raches, dans le Nord, sont identiques au niveau des deux premiers mots. À un moment ou à un autre de l'arborescence, il est *certain* que des bigrammes du 3e mot seront employés, car ce sont les seuls à permettre de discriminer entre les deux communes. Au pire, s'il y a un seul bigramme qui diffère entre deux libellés de codes distincts, on est sûr, avec cette technique, qu'il sera exploré.

1.3.2.2. Questionnement

Étant donné un libellé à coder, *on va d'abord le normaliser* : traiter les caractères blancs et vides, synonymiser, éliminer les mots vides, structurer le libellé (département + 4 mots de 12 caractères dans le cas qui précède).

À la racine de l'arbre de questionnement figurera l'information selon laquelle la position de bigramme à étudier est la 2e du 1er mot ; l'on extraira donc le deuxième bigramme du premier mot de notre libellé, puis on recherchera s'il existe une branche qui correspond.

Arrivé au noeud suivant, on réitère la même opération : ce noeud est associé à une position de bigramme, que l'on va extraire de notre libellé, pour ensuite déterminer s'il existe une branche qui correspond.

Ce cheminement n'est pas infini ; il se termine dans l'une des situations suivantes :

- 1. *Le noeud considéré est une feuille de l'arbre*, il correspond à un code (cf. l'explication de Verrières-le-Buisson) : ce code est celui que nous donnerons à notre libellé.

Remarque : il peut également s'avérer que dans un fichier d'apprentissage, on retrouve un même libellé plusieurs fois, avec des codes différents ; la feuille de l'arbre représente donc plusieurs codes possibles ; on parle alors d'*indécision* (dans Sicore, on utilise plutôt le terme de *codage multiple*).

- 2. *Il n'y a pas de branche correspondant à la valeur de bigramme que l'on trouve dans le libellé* (exemple : on doit coder "92 VERRIERES BUISSON") ; dans ce cas, le système se dit incapable de coder, il y a échec du chiffrement.

1.3.2.3. Les bigrammes de redondance

Les divers tests qui ont été effectués avec Quid ont montré que l'algorithme pouvait parfois coder en se fondant sur très peu de bigrammes, parfois même un seul. Par

exemple, dans le cas du chiffrement de la commune, le fait que le deuxième bigramme du premier mot ait pour valeur "II" conduit immédiatement à la commune de Briis-sous-Forges, dans l'Essonne.

À partir de là, dans un libellé à coder, le moindre doublement d'un "I" placé en troisième lettre va conduire automatiquement à coder ce libellé comme Briis-sous-Forges, et ce *quels que soient les autres caractères du libellé.* Le mode de fonctionnement de l'algorithme, on le voit donc, a quelques conséquences peu souhaitables : certains libellés sont codés alors qu'ils ne devraient pas l'être.

D'où l'idée d'effectuer un *contrôle sur les libellés codés*, consistant à vérifier s'il y a égalité entre le libellé codé et le libellé du fichier d'apprentissage, au niveau de quelques bigrammes jugés importants. Ce contrôle est appelé **contrôle de redondance**.

Pour le faire, on se donne d'abord une liste de positions de bigrammes, appelés **bigrammes de redondance** : en général, les 2 ou 3 premiers bigrammes des premiers mots.

Le codage se déroule alors en 2 temps :

- *questionnement* par Quid, selon la méthode exposée plus haut ;

- lorsqu'un code est obtenu, *vérification* de l'exactitude des bigrammes de redondance du libellé.

Cette technique augmente nettement la qualité de codage, mais détériore (logiquement) son efficacité.

1.3.3. Critères de qualité

Pour comparer entre eux des programmes de codage automatique, il faut disposer de critères de qualité. Il est assez facile de les faire émerger : pour "bien" coder automatiquement, il faut coder beaucoup, coder juste et coder vite.

On utilise donc les indicateurs suivants :

- **efficacité** = pourcentage de libellés codés automatiquement parmi un ensemble donné de libellés ;

- **fiabilité** = pourcentage de "bien codés" parmi les libellés codés ;

- **temps unitaire de codage** = temps moyen de codage d'un libellé.

On peut constater que ces critères n'ont rien d'absolu : ils dépendent entièrement du fichier que l'on a cherché à chiffrer automatiquement. C'est sans doute le troisième critère qui sera le plus stable, *à programme donné*, et sur lequel on pourra sans doute le moins agir.

En revanche, il est possible d'intervenir sur les deux premiers critères, qui ont le désavantage d'être antagonistes. En effet, on peut augmenter l'efficacité en jouant sur les paramètres de l'algorithme (par exemple, en relâchant les seuils, dans le cas des mesures de distance) ; mais la conséquence est alors immédiate : la fiabilité chute. À l'inverse, on peut être plus sévère, plus exigeant sur la proximité entre libellés à coder et libellés officiels : la fiabilité croît, mais l'efficacité s'en ressent.

Si l'on suppose fixées les connaissances nécessaires au codage (nomenclature, par exemple), une des grandes difficultés du chiffrement automatique est d'arbitrer entre efficacité et fiabilité. En pratique, cela signifie qu'il faut se donner des seuils (fiabilité minimum, par exemple), et disposer d'instruments pour effectuer cet arbitrage.

2. Le codage automatique dans Sicore

2.1. Principes généraux

La méthode Quid conduisait à une externalisation des connaissances en séparant le fichier d'apprentissage des programmes capables de le gérer. Les programmes de questionnement étaient donc généraux : il s'agissait d'un simple parcours de l'arborescence générée par l'apprentissage.

Sicore étend ce principe : toutes les informations nécessaires au codage d'une variable donnée se trouvent dans des *bases de connaissances*, séparées des programmes généraux. Le fichier d'apprentissage est l'une de ces connaissances. Grâce à cette séparation programmes- connaissances, les programmes sont toujours les mêmes quelle que soit la variable à coder.

Pour coder automatiquement avec Sicore, il n'y a donc que deux opérations : charger les connaissances sur la variable, puis coder. Insistons bien sur la première étape ; sans connaissances, Sicore est une coquille vide, incapable du moindre chiffrement automatique. Il faut donc l'alimenter en connaissances avant de faire quoi que ce soit.

L'expression "codé automatiquement par Sicore" est donc, en toute rigueur, impropre[1] : **un codage automatique est toujours réalisé par un couple (Sicore , bases de connaissances).**

5. Mais on l'emploie tout de même, pour simplifier les notations.

Cette remarque a une conséquence importante : lorsque le résultat du codage automatique est faux, Sicore n'est pas, en général, responsable de cette erreur : celle-ci figure beaucoup plus souvent dans les bases de connaissances qu'il utilise.

2.2. *Les bases de connaissances*

Une base de connaissances est un ensemble cohérent de connaissances sur un domaine donné, et surtout *relativement à une variable donnée*. Ces connaissances doivent être écrites de façon déclarative et non impérative : il ne s'agit pas de programmes que l'on applique, mais de structures complexes que l'on peut utiliser de diverses manières.

Dans Sicore, il y a **six bases de connaissances sur le codage**, autrement dit six domaines qui résument l'expertise ; ces bases décrivent :

- le passage d'un texte à un autre texte plus facilement traitable (**règles de normalisation**) ;
- le passage de l'univers des textes à l'univers des codes (**fichier d'apprentissage**) ;
- le passage d'un code "inutilisable" à un autre code, "utilisable", via les variables annexes (**règles logiques de codage**) ;
- le **paramétrage de l'algorithme** d'apprentissage ;
- le passage d'une valeur de variable annexe à une autre valeur, faisant sens pour les règles logiques (**transcodages**) ;
- la structure des enregistrements à coder (**dessins de codage**).

2.2.1. Les règles de normalisation

Elles décrivent comment fabriquer, à partir d'un libellé brut, un libellé propre, adapté au questionnement. L'utilisateur y définit la liste des caractères blancs, des caractères vides, des mots vides et des couples synonymes, ainsi que les règles de calibrage.

Au début de l'étape de normalisation, le libellé brut n'est pour Sicore qu'une succession de caractères ; les caractères vides (le point, par exemple), sont purement et simplement éliminés. Les caractères blancs (l'apostrophe, le tiret, ...) sont remplacés par des blancs, et jouent donc le rôle de séparateurs. Dès lors, il ne reste plus qu'à éliminer les mots vides (articles et prépositions, en général), puis à remplacer les mots du libellé par leurs synonymes (exemple : ADJT = ADJOINT), lorsqu'ils en ont. Dans Sicore, il est également possible de travailler sur des groupes de mots (exemple : ELECTRICITE

DE FRANCE = EDF), ce qui oblige à faire attention à l'ordre des mots vides et des synonymes, qui influe sur le résultat de la normalisation.

Le *calibrage* consiste alors à donner au libellé obtenu une longueur fixée, en limitant le nombre de mots et la taille des mots (exemple : 5 mots de 10 caractères). Lorsque les mots sont trop courts ou trop peu nombreux, on complète par des blancs. Le fait que le libellé soit calibré permet de le rendre apte à passer par l'arbre de questionnement, après décomposition du texte en bigrammes.

2.2.2. Le fichier d'apprentissage

Il demeure la base de connaissances la plus importante. Il possède une structure très simple : c'est un fichier dans lequel on trouve, sur chaque ligne, un libellé et un code. Lorsque le libellé est trop imprécis, et que l'ambiguïté peut être levée par des variables annexes, le code en question n'est parfois qu'un code intermédiaire, désignant une *table de décision.*

Ainsi, l'on ne peut pas associer au libellé de profession "DESSINATEUR" un code PCS, car cela dépend de plusieurs informations supplémentaires : statut, qualification ... On lui fait donc correspondre une table de profession (qui s'appelle, en l'occurrence, T-Y386). Cette table de profession est elle-même définie dans une autre base de connaissances, le fichier des règles logiques.

L'utilisateur de Sicore manipule deux types de fichiers d'apprentissage : le fichier d'apprentissage brut (FAB) et le fichier d'apprentissage (FA), dans lequel ces libellés sont normalisés. **On fabrique le FA en normalisant le FAB** ; cette normalisation emploie bien évidemment la base de connaissances évoquée précédemment, à savoir les règles de normalisation. Ce mode de fonctionnement assure une cohérence parfaite entre ces règles et l'arbre de questionnement. En pratique, donc, l'utilisateur ne *modifie jamais le FA* : il modifie le FAB, le normalise, et le FA ne lui sert qu'au moment de l'apprentissage.

Normaliser le FAB, c'est passer de :

Figure n° 5 :

Extrait d'un fichier d'apprentissage brut (Fab) de départements

01	Ain
02	Aisne
03	Allier
04	Basses-Alpes
04	Basses-Al
04	Bass. Alp.
04	BassesAlpes
05	Hautes-Alpes
05	Hautes-Al

05	HautesAlpes
05	Haut. Alp
06	AlpesMaritimes
06	Alpes-Maritimes
06	Alpes-Maritim
06	Alpes-Marit

à :

Figure n°6 : extrait d'un FA de départements, normalisation du FAB[1] précédent

AIN		01
AISNE		02
ALLIER		03
BASSES	ALPES	04
BASSES	AL	04
BASS	ALP	04
BASSESALPES		04
HTES	ALPES	05
HTES	AL	05
HAUTESALPES		05
HT	ALP	05
ALPESMARITIM		05
ALPES	MARITIMES	06
ALPES	MARITIM	06
ALPES	MARIT	06

2.2.3. Les règles logiques

Elles ne servent que si le libellé seul ne suffit pas pour coder, d'autres informations permettant de lever l'ambiguïté. Dans ce cas, comme on l'a vu, le texte en question est associé à une *table de décision*. Le fichier des règles logiques est celui qui contient la définition de toutes les tables de ce type.

Une table de décision est une succession de règles logiques, faisant intervenir des variables annexes.

Prenons un exemple simple : pour coder la commune, on utilise l'année comme variable annexe. Ainsi, la commune d'Asnières, dans les Hauts-de-Seine, a changé de code au cours du temps : il valait 75001 jusqu'à 1964, il vaut 92004 depuis. Dans le fichier d'apprentissage, on va donc associer au libellé "92 ASNIERES" un code qui ne sera ni 75001, ni 92004, mais une table de décision, que l'on peut nommer comme on veut : par exemple, 92ASN.

6. On a utilisé, dans cette normalisation : un calibrage de 3 mots de 12 caractères ; le code département comme information passive ; des caractères vides comme le tiret ; des synonymes comme HAUT=HT.

La table "92ASN" sera alors définie comme suit :

Figure n°7 : un exemple de table de décision

```
92ASN  ANNÉE ;
REGLE > 1964 ; 92004;
REGLE ;          75001;
```

Remarquons simplement que les règles logiques ont en général une structure plus complexe que celles-ci, tout simplement parce que les variables annexes sont plus nombreuses : 11 pour la profession, 6 pour les libellés d'occupation.

2.2.4. Les paramètres de l'algorithme d'apprentissage

Il s'agit d'une base de connaissances de petite taille, qui permet à l'utilisateur d'agir sur la façon dont l'arbre de questionnement sera construit, ce qui peut conduire à des gains substantiels en temps d'apprentissage et en efficacité de codage, comme on le verra plus loin. En pratique, ce paramétrage n'est pas immédiat à mettre au point, et nécessite une bonne connaissance du fonctionnement de Sicore.

2.2.5. Les connaissances spécifiques à l'enquête

Les **transcodages** et les **dessins de codage**, à l'inverse des bases de connaissances précédentes, n'ont pas un caractère général : *elles sont spécifiques à la source* qui doit être codée.

Un *dessin de codage* est en quelque sorte un dessin de fichier : on y définit où se trouvent les informations : libellé, variables annexes, informations "passives", ... On y explique aussi comment déplacer ces informations, si nécessaire (ce qui revient à un dessin du fichier résultat).

Les *transcodages* décrivent de quelle manière "traduire" les valeurs de variables annexes. En effet, au sein des règles logiques, toute variable annexe prend ses valeurs dans une certaine liste (1, 2, et 3 pour le statut), mais il se peut que les notations soient différentes dans l'enquête (par exemple a, b, c), voire que la variable prenne plus de valeurs.

Toutes ces bases de connaissances peuvent être chargées, apprises (dans le cas du fichier d'apprentissage) et regroupées en un seul fichier, appelé **fichier d'environnement**. Celui-ci contiendra l'arbre de questionnement, les règles de normalisation, le tout écrit dans une syntaxe interne que reconnaît Sicore.

2.3. *L'algorithme d'apprentissage de Sicore*

L'algorithme d'apprentissage de Sicore se fonde, mathématiquement, sur l'algorithme Quid. Sicore, écrit en C, n'utilise strictement aucun programme de Quid (écrit en PL/1 et en Cobol), mais la programmation de l'algorithme d'apprentissage a repris strictement, tout du moins au départ, l'algorithme théorique originel.

Au fur et à mesure de l'avancement du projet Sicore, de nombreuses *variantes* y ont été néanmoins apportées. **Leur but était toujours le même : permettre à l'utilisateur d'agir sur la constitution de l'arbre de questionnement, en d'autres termes "assouplir" le fonctionnement de l'algorithme originel.**

2.3.1. Spécificités de l'algorithme

L'issue de l'algorithme Sicore est toujours un arbre de questionnement, obtenu à partir d'un fichier d'apprentissage normalisé ; les libellés y sont toujours décomposés en bigrammes[1]. *L'arbre obtenu a une structure un peu différente de l'arbre Quid*, car les bigrammes de redondance n'y sont pas représentés de la même manière : clé de redondance dans Quid, extension de l'arbre dans Sicore ; mais cette différence ne joue que sur les temps d'apprentissage et de codage, et non sur les résultats eux-mêmes.

Sicore donne à l'utilisateur la possibilité de :

- se doter d'une liste de bigrammes prioritaires, c'est-à-dire fixer, dans une première phase de la construction de l'arbre, un ordre de choix de bigrammes ;
- imposer qu'un bigramme ne soit pas pris en compte avant une certaine profondeur de l'arbre ;
- imposer qu'un bigramme ne soit jamais pris en compte avant un autre bigramme ;
- assouplir le contrôle de redondance, en "acceptant" un bigramme s'il est constitué de deux blancs, quelle que soit la valeur du bigramme auquel on le compare ;
- introduire des mots "jokers" dans le fichier d'apprentissage.

C'est la technique des **bigrammes prioritaires** qui s'est révélée la plus utile. C'est pourquoi nous ne détaillerons que celle-là.

1. Comme dans QUID, on peut utiliser des trigrammes, des monogrammes, des quadrigrammes, mais il s'avère que le bigramme est le meilleur choix.

Nous avons vu que dans son principe même, Quid s'appuyait sur la notion de critère d'information *local* : à chaque niveau de l'arbre, on recherche la position de bigramme apportant le plus d'information.

La notion de bigramme prioritaire est presque à l'opposé de cela : on va donner au système d'apprentissage une liste de positions de bigrammes, dont l'ordre n'est pas indifférent, et l'arbre commencera à être construit selon ces positions et dans cet ordre, jusqu'à ce que la liste soit épuisée. *La suite de la construction de l'arbre, quant à elle, reprendra le principe de critère d'information local* ; ceci signifie que le critère local n'est nullement abandonné : il apparaît seulement un peu plus loin dans l'arbre.

Ainsi, si la liste de bigrammes prioritaires est (2,1,8), on commencera par la seconde position de bigramme, et que l'on poursuivra avec la première et la huitième[1] ; c'est seulement à partir de ce moment-là (i.e. après avoir exploré trois noeuds de l'arbre) que l'on calculera, si nécessaire, des critères d'information pour chercher le bigramme le plus informant.

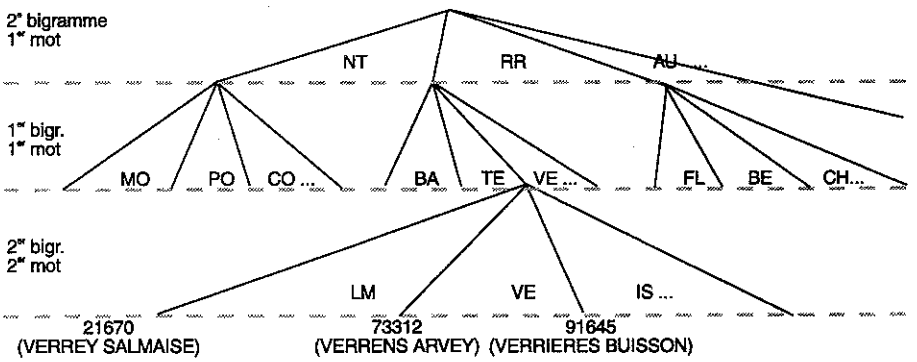
D'où une arborescence qui a l'allure suivante (on repart du même fichier d'apprentissage que précédemment) :

Dans le cas de la commune, les bigrammes 2, 1 et 8 étant connus, on trouve ainsi en trois étapes les communes de Verrey-sous-Salmaise, Verrens-Arvey et Verrières-le-Buisson (les "-" et le mot "sous" disparaissent après normalisation).

Si l'on compare avec l'arbre Quid, on s'aperçoit qu'ici, *aux trois premiers niveaux de l'arbre*, les positions de bigrammes prises en compte sont **fixes**, alors qu'elles variaient dans le cas précédent. On y trouvait en effet à la même profondeur de l'arbre le premier bigramme du premier mot, le département, et le premier bigramme du deuxième.

Figure 8 :

Arbre de questionnement avec les bigrammes prioritaires 2, 1, 8



1. Le bigramme du 2e mot, dans le cas de mots de 12 caractères.

On remarquera également que le fait que le deuxième bigramme du premier mot soit placé en première priorité ne change rien au début de l'arbre, relativement au premier schéma : en effet, c'était ce bigramme qui apportait le plus d'information.

2.3.2. Avantages des bigrammes prioritaires

En premier lieu, cette technique conduit à une forte **diminution des temps d'apprentissage**, puisqu'il n'est plus nécessaire de calculer toutes les entropies conditionnelles, qui obligeaient à de nombreux tris, coûteux en temps. Lorsqu'on utilise, comme bigrammes prioritaires, les deux premiers bigrammes des deux premiers mots (ce que l'on fait la plupart du temps), on divise en général le temps par 3. Le gain peut même aller jusqu'à un facteur 4, voire 5.

Le gain en temps d'apprentissage est encore plus important vis-à-vis du *logiciel* Quid : dans le cas d'un fichier d'apprentissage d'environ 150 000 libellés, l'apprentissage par Quid prenait environ 3 h 20 CPU[1], alors qu'elle nécessitait (avec bigrammes prioritaires) environ 1'30'' CPU avec Sicore[2]. Ceci représente **un écart en temps de 1 à 100,** qui n'est sans doute pas l'écart exact (il faudrait plus d'exemples), mais donne néanmoins un ordre de grandeur.

On constate également que **les temps de codage ne sont pas détériorés** : lorsqu'on utilise des bigrammes prioritaires, le temps moyen de codage est équivalent à celui que l'on obtient avec un arbre basé entièrement sur le critère d'information local. Ces temps sont de l'ordre de quelques dix millièmes de seconde CPU sur le site du CNI de Paris. Le temps de codage n'est donc pas réellement un problème pour Sicore.

Les tests sur la raison sociale - adresse (pour le prochain RP) ont également montré que l'ajout de bigrammes prioritaires conduisait à une **augmentation conséquente de l'efficacité**. Dans ce cas particulier, on a en effet constaté qu'il était très avantageux de choisir des bigrammes d'adresse pour premiers bigrammes prioritaires. Ce gain en efficacité a également été mis en évidence sur des libellés de profession provenant des DADS : on trouve un écart de 4,8 points entre un Sicore avec choix raisonné de bigrammes prioritaires et un Sicore sans bigrammes prioritaires (simili-Quid).

On remarque aussi qu'avec l'utilisation de plusieurs bigrammes prioritaires, **l'arbre Sicore est nécessairement plus stable que l'arbre Quid** : les premières branches de l'arbre, qui correspondent aux bigrammes prioritaires, ne peuvent pas être modifiées par l'introduction de nouveaux libellés dans le fichier d'apprentissage. Or cette instabilité lors des évolutions du fichier d'apprentissage était l'un des défauts majeurs de Quid.

9. Sur le site du CNI d'Orléans, à l'été 1994.
10. Une petite part de ce gain est due à l'accroissement de la puissance de la machine du CNI de Paris (qui a eu lieu en octobre 1994).

Enfin, les bigrammes prioritaires[1] font partie **des instruments qui vont permettre à l'utilisateur de gérer son arbitrage efficacité-fiabilité.** En modifiant ceux-ci, ainsi que les autres instruments (bigrammes de redondance non prioritaires, bigrammes pour lesquels on assouplit le contrôle de redondance), l'utilisateur va rechercher le paramétrage optimal, qui lui permettra d'atteindre la meilleure efficacité possible tout en conservant une fiabilité acceptable. Une telle opération requiert un minimum d'expertise de sa part.

2.4. L'algorithme de codage

2.4.1. La méthode de base

Une fois que Sicore a été grassement nourri de connaissances sur le chiffrement (entre autres l'arbre de questionnement), il devient apte à coder, mais cette opération constitue une boîte noire pour l'utilisateur. Celle-ci fonctionne en fait en **trois étapes.**

La **normalisation** consiste à éliminer les caractères blancs ou vides, les mots vides, à remplacer les synonymes, et à calibrer le libellé. Le **questionnement**, à partir du libellé normalisé, n'est autre que le parcours de l'arborescence pour trouver un code, qui peut n'être qu'un code intermédiaire (même principe que Quid). Dans ce dernier cas, on passe par une étape de **détermination du code final grâce aux variables annexes** et aux règles logiques de codage.

À partir d'un fichier de libellés à coder, Sicore fournit alors un fichier résultat, qui a une structure connue. Dans celui-ci, on trouve, à chaque ligne : un libellé normalisé, les informations passives que l'on a souhaité conserver, un écho de codage, et enfin un résultat de codage (0, 1 ou plusieurs codes).

L'écho de codage permet de savoir comment s'est passé le codage, en répondant aux questions du genre : le codage a-t-il abouti ? sinon, pourquoi ?

1. Qui, dans Sicore, sont obligatoirement des bigrammes de redondance.

Cet écho peut prendre différentes valeurs[1], décrivant les situations suivantes :

Écho de codage	Signification
Codage simple	Le libellé a été codé, sans utiliser de variables annexes, et il a subi avec succès l'épreuve du contrôle de redondance.
Erreur de redondance	Comme dans le cas précédent, le questionnement a abouti, mais le contrôle de redondance n'a pas été satisfaisant. On considère donc le libellé comme étant non-codé.
Échec de codage	Le questionnement n'a pas abouti, ce qui signifie qu'à un noeud de l'arbre, la branche correspondant à la valeur du bigramme disponible n'était pas présente.
Codage multiple	Il y avait, dans le fichier d'apprentissage, au moins un libellé associé à plusieurs codes différents ; par conséquent, lors de la création de l'arbre, une feuille de celui-ci a été associée à plusieurs codes. Au moment du codage, Sicore renvoie donc, logiquement, les codes en question. Dans ce cas-là, on considère également que Sicore a échoué.
Codage simple en passant par les règles logiques	Le libellé a été reconnu, le contrôle de redondance a été bon, et l'on a abouti à un code en passant par les règles logiques, sans aucune erreur dans cette dernière phase.
Erreur en parcourant les règles logiques	Le libellé a été reconnu, le contrôle de redondance a été bon, mais dans au moins une table de décision, aucune des règles n'était applicable, et il n'a donc pas été possible de poursuivre le parcours des règles jusqu'à un code final[2].

Un libellé est donc considéré comme codé automatiquement si et seulement si l'écho de codage correspond à un *codage simple* ou à un *codage simple en passant par les règles logiques.*

2.4.2. Codage de libellés hétérogènes

La technique précédente n'autorise pas le chiffrement automatique dans le cas de libellés composés de plusieurs sous-libellés distincts : les *libellés hétérogènes.*

2.4.2.1.Codage complexe

Si l'on veut effectivement coder un enregistrement hétérogène, il est nécessaire de procéder autrement, **en agissant séparément sur les diverses composantes du libellé.** On nomme **codage complexe** un tel mode de chiffrement, que l'on peut décrire comme une succession de normalisations et de codages sur des composantes du libellé.

12. On ne décrit ici que l'essentiel.
13. Il est à noter que ce cas de figure est souvent voulu par l'expert qui a créé les bases de connaissances : ceci permet, en particulier, de gérer les incohérences entre libellé et variables annexes.

Exemple 1 : libellé de département + libellé de commune.

Pour traiter l'ensemble, on commence par coder le département, en considérant la commune comme une information passive que l'on veut conserver, puis, dans un deuxième temps, on code l'ensemble code département + libellé de commune.

Exemple 2 : raison sociale - adresse

La situation est encore plus difficile : on dispose d'un libellé fortement hétérogène contenant une raison sociale, un département, une commune, un type de voie, une voie. Il faut donc coder la commune, normaliser chaque partie du libellé, jusqu'à ce que l'on aboutisse à une raison sociale - adresse normalisée. Ce traitement sera le même pour le fichier d'apprentissage brut et pour les libellés à coder ; ce sont les libellés ainsi normalisés qui passeront ensuite dans l'arbre de questionnement.

2.4.2.2. Codage multi-arbres

Lorsqu'on travaille sur des libellés hétérogènes, on peut aussi améliorer l'efficacité du chiffrement en appliquant plusieurs arbres de questionnement basés sur des principes différents. L'idée est la suivante : dans la mesure où le libellé est hétérogène, il se peut que telle ou telle partie du libellé soit absente, ou de mauvaise qualité, alors que le reste est bon ; on peut donc essayer de fabriquer un arbre dans lequel telle partie du libellé est "privilégiée" (grâce aux bigrammes prioritaires), puis un autre privilégiant une autre composante.

Ainsi, lorsqu'on code un SIRET à partir d'une raison sociale-adresse, on peut décider de se focaliser sur la raison sociale et la commune (en ne plaçant aucun bigramme de redondance sur la voie), ou bien s'appuyer au contraire sur la voie, en limitant le nombre de bigrammes de redondance de la raison sociale. Cette technique a été appliquée avec succès sur des intitulés du précédent RP, dans le département du Nord : en utilisant le premier arbre de questionnement, on obtenait une efficacité de 37,5%[1] ; avec le deuxième, appliqué aux non-codés, l'efficacité combinée passait à 49%. Le **codage multi-arbres** permet donc ici de *gagner 11 points en efficacité*. Ce mode de codage n'est pas indissociablement lié aux libellés hétérogènes, mais c'est dans ce domaine qu'il est sans doute le plus approprié.

1. L'efficacité de codage du SIRET est de toutes façons très mauvaise pour de nombreuses raisons : complexité du libellé, difficulté de mettre à jour le fichier d'apprentissage (plusieurs millions d'établissements), utilisation de "noms courants", ...

3. Le système Sicore

Pour qu'une application de codage automatique demeure efficace à long terme, il faut qu'elle soit **vivante** : le langage évolue, de nouvelles expressions apparaissent, et les nomenclatures changent également (apparition de nouveaux métiers, fusion de communes, nouvelle façon d'appréhender les activités...). Si ce n'est pas le cas, les "codeurs" qui traitent les libellés non-codés automatiquement récupèrent à chaque fois les mêmes mots, les mêmes termes, ce qui entraîne une démotivation de leur part.

Ceci signifie que **les connaissances doivent évoluer de façon régulière** : il faut, sans cesse, réalimenter les bases de connaissances en expressions nouvelles, en codes nouveaux ; on doit aussi mettre en évidence de nouvelles variables annexes, des règles logiques plus détaillées, des synonymes utiles, des mots vides opportuns.

Par conséquent, un codage automatique qui se veut réellement général doit se préoccuper non seulement de l'existence de bases de connaissances solides, mais aussi de la **mise à jour** de celles-ci, ce qui implique une réflexion sur la **méthodologie** et l'**organisation** qui accompagnent cette évolution.

Cette constatation a d'ailleurs été à la base du "projet Sicore".

3.1. La boucle Sicore

Il est clair que la mise à jour des bases de connaissances doit prendre appui sur les codages qui ont déjà eu lieu. Ce qui n'est pas codé, ce qui est mal codé doit pouvoir être réanalysé. Et ce sont justement ces erreurs, ces échecs, qui, une fois pris en compte, permettront de rendre plus corrects et plus efficaces les chiffrements automatiques ultérieurs. Une telle **rétroaction** peut avoir lieu à l'issue de tout codage automatique.

Le fonctionnement envisagé dans le cadre de Sicore est ainsi fondé sur un principe de **feed-back permanent** : codage automatique analyse des résultats mise à jour des connaissances et vérification de cohérence création d'un nouveau fichier d'environnement codage automatique, basé sur ce fichier d'environnement.

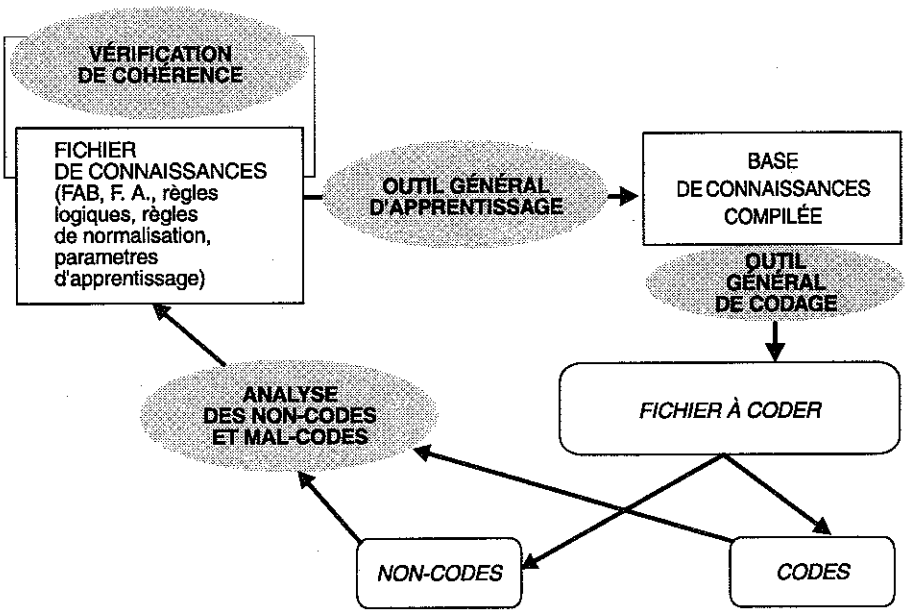
Il s'agit donc d'un processus cyclique qui s'auto-entretient, s'**autocatalyse**, car chaque opération du processus crée les conditions permettant à l'opération suivante d'être exécutée : c'est grâce aux non-codés et mal-codés que l'expert pourra réévaluer ses bases, c'est grâce à la modification qu'il y aura un "nouveau" codage automatique.

Le tout conduit donc à **un enrichissement régulier des connaissances**, et parallèlement à une amélioration tout aussi régulière du codage automatique.

Le cycle en question, appelé **boucle Sicore**, a l'allure suivante :

Figure n°9 :

Boucle Sicore (pour une variable donnée)



Dans ce schéma (où les ellipses sont les outils, et les rectangles, les données), la partie droite correspond à la production, et le reste au travail d'expertise.

Pour que le codage automatique fonctionne, il faut[1] une "base de connaissances compilée", appelée également **fichier d'environnement** : ce fichier contient toutes les connaissances sur le codage, et en particulier l'arbre de questionnement.

À l'issue du codage automatique, **l'expert analyse les libellés non-codés, mais aussi un échantillon de libellés codés**, ce qui lui permet de trouver quelques enregistrements qui ont été "mal" codés : c'est l'expert qui juge de ce qui est "bien" ou "mal" codé.

En fonction de ce qu'il a découvert, **il modifie ses bases de connaissances** : il ajoute des libellés dans le fichier d'apprentissage brut, modifie des règles logiques, introduit

1. C'est même nécessaire *et* suffisant.

de nouveaux synonymes, vérifie les transcodages ... Rappelons que les paramètres d'apprentissage, et en particulier les *bigrammes prioritaires*, font partie de ces connaissances.

Vient ensuite l'étape de **contrôle de cohérence** : il s'agit d'une boucle interne à la boucle de reformulation. Lorsque les nouvelles bases de connaissances ont été mises au point, on vérifie leur cohérence ; s'il y a problème, ceci conduit à des modifications, qui peuvent elles-mêmes affecter la cohérence en d'autres points. On poursuit ainsi jusqu'à ce que l'ensemble des connaissances soit jugé correct. Dès lors, on crée un fichier d'environnement, qui sera utilisé pour un nouveau chiffrement automatique, et on entame ainsi une nouvelle boucle.

3.2. Mise au point des connaissances

La mise au point des connaissances est une opération lourde, dans la mesure où les bases de connaissances (notamment le fichier d'apprentissage, les règles de normalisation et les règles logiques) sont de taille importante. **Le but de cette opération est clair : il s'agit d'augmenter la qualité des bases de connaissances**. Or il existe deux critères essentiels de qualité[1] : l'**efficacité**, i.e. le taux de codage automatique, et la **fiabilité**, i.e. le pourcentage de bien codés parmi les codés. En mettant à jour les connaissances, on cherche à améliorer l'un ou l'autre de ces deux critères, et si possible les deux.

Si l'on part d'une situation où un fichier vient d'être codé automatiquement, elle se déroule en trois temps : détection de problème, modification, validation.

3.2.1. Détection de problème

Dans la *phase de détection*, on recherche comment améliorer les bases de connaissances en analysant d'une part les non-codés, d'autre part les mal-codés. Pour cela, il convient de **visualiser correctement les fichiers**, en mettant en évidence les problèmes les plus importants : le TFD (Tri par Fréquence Décroissante) est un outil de base pour cela, car il permettra de ne se polariser que sur les difficultés les plus fréquentes, qui ont donc une influence sur l'efficacité.

On peut appliquer un TFD aux libellés complets (pour trouver ce que l'on va ajouter au fichier d'apprentissage), mais aussi aux premiers mots du libellé, ce qui permettra de mettre en évidence des mots vides ou des synonymes utiles. À titre d'exemple,

16. En effet, le troisième critère évoqué plus haut, à savoir le temps unitaire de codage, ne joue pas beaucoup pour Sicore : d'une part en raison de la rapidité de Sicore, d'autre part parce que ce temps de codage est très stable.

lorsqu'on effectue un TFD sur des libellés d'occupation non-codés, on trouve que le mot "JE" est extrêmement fréquent ; comme il n'apporte pas vraiment d'information, on pourra l'ajouter à la liste des mots vides.

L'analyse des mal codés se fera autrement : un principe possible est de **tirer un échantillon à partir de l'ensemble des libellés codés**, en pondérant chaque libellé par sa fréquence d'apparition dans le fichier[1]. Dès lors, tous les intitulés de l'échantillon seront étudiés un par un, ce qui fera émerger un (petit) groupe de libellés mal codés.

De façon plus générale, l'analyse des résultats de codage dans Sicore se fait en utilisant le concept de **transformation**. On nomme "transformation" une application qui transforme un fichier en un autre fichier, résultant de la composition d'*opérations élémentaires* figurant parmi les sept suivantes : tri, simplification, filtrage, aménagement, tirage d'échantillon, normalisation, codage.

Les termes tri, tirage d'échantillon, normalisation, codage, sont clairs. La simplification consiste à remplacer toute occurrence multiple d'un libellé par le libellé seul, précédé éventuellement par la valeur de la fréquence de celui-ci. Le filtrage a pour paramètre un "filtre", c'est-à-dire une proposition logique : il a pour résultat le fichier de tous les libellés satisfaisant la proposition logique. L'aménagement consiste à déplacer ou à éliminer des champs sur chaque ligne du fichier initial.

Donnons-en quelques exemples.

Le tirage d'un échantillon de mal-codés correspond à la composition de 3 opérations élémentaires : codage, tirage d'échantillon, filtrage.

Un tri par fréquence décroissante des deux premiers mots parmi les libellés non-codés résulte de la transformation suivante : filtrage (non-codés) + aménagement (deux premiers mots) + tri + simplification + tri

3.2.2. Modification de connaissances

La *phase de modification* consistera à ajouter, ôter ou modifier des connaissances dans une ou plusieurs bases. En réalité, toutes les bases ne sont pas concernées : en pratique, les paramètres d'apprentissage, dessins de codage et transcodages sont rapidement fixés. On enrichira donc le FAB de quelques intitulés, on ajoutera ou on ôtera des synonymes, on affinera quelques règles logiques.

1. On aura évidemment fait en sorte que chaque libellé n'apparaisse qu'une fois (avec sa fréquence) dans le fichier en question.

Tout le problème de la phase de modification est de savoir quelles connaissances mettre à jour, et jusqu'à quel point ; en particulier, il y a souvent hésitation entre ajout de synonyme ou ajout de libellé dans le Fichier d'Apprentissage Brut. C'est pourquoi ce point mérite d'être tout particulièrement détaillé.

3.2.2.1. Choix entre synonyme et FAB : cas où l'on ne modifie rien

Dans ce cadre, la technique des bigrammes prioritaires permet de faciliter le travail, notamment si l'on a choisi les premiers des premiers mots : en effet, on est sûr que les "racines"[1] des mots sont systématiquement prises en compte ; or ces racines suffisent souvent à la reconnaissance, et il est alors inutile d'ajouter des synonymes de même racine.

Par exemple, si l'on prend un mot comme "ABATTEUR", il ne sera pas nécessaire d'écrire que "ABATT", voire "ABAT" sont des synonymes de ce mot : si l'on regarde tous les libellés de profession ayant un mot commençant par "ABAT", on s'aperçoit en effet que ces 4 premiers caractères suffisent.

Conclusion : *si la racine suffit à comprendre le libellé, il n'est besoin ni de rajouter des libellés dans le FAB* (exemple : ABATT VOLAILLES en plus de ABATTEUR VOLAILLES), *ni de rajouter des synonymes* (comme ABATT = ABATTEUR).

Dans le cas où la racine de 4 lettres[2] ne suffit pas, on peut quand même tirer du fonctionnement de l'algorithme une autre recommandation : *si un mot peut être décliné de multiples façons à partir d'une même racine*[3] *longue* (ex: TELEPROS, TELEPROSPECTEUR, TELEPROSPECT, TELEPROSPEC, TELEPROSP), *il faut que, d'une manière ou d'une autre, cette racine figure dans le FA*. Si ce point est acquis, les synonymes ne sont, à la limite, même pas nécessaires. Si la racine est courte, les synonymes deviennent indispensables.

Remarquons au passage que l'on a un moyen général simple pour voir sur quels libellés les racines ne permettent pas de coder : on fait un apprentissage dans lequel on ne prend en compte que deux mots de 4 lettres (c'est-à-dire que l'on redéfinit complètement les paramètres d'apprentissage, le FA restant le même mais n'étant pas "appris" de la même façon) ; puis on recherche les codages multiples, qui risquent d'être nombreux.

1. On pratique, il s'agit souvent des 4 premiers caractères, pour les deux premiers mots.
2. Ou bien 6, si les trois premiers bigrammes du mot sont prioritaires.
3. Remarque générale sur les racines, courtes ou longues : on raisonne sur des racines ayant un nombre entier de bigrammes, donc un nombre pair de caractères.

3.2.2.2. Choix entre synonyme et FAB : ajout de libellés dans le FAB[1]

Une fois les bigrammes prioritaires parcourus, Sicore recherche le bigramme apportant le plus d'information, selon le critère de Quid. **Ce critère de choix est local** : selon les valeurs de bigrammes déjà rencontrées, on va aller chercher le premier bigramme du quatrième mot, ou le cinquième bigramme du premier mot, ... De par le fonctionnement même de l'algorithme, le bigramme choisi devra apporter de l'information. *Ainsi, si deux libellés ont des mots différents de racines identiques, et des codes distincts, il y a des chances pour que les bigrammes ultérieurs (post racine) des mots soient utilisés pour les départager.* C'est même certain pour des libellés d'un seul mot (comme "AIGUILLEUR" et "AIGUISEUR").

Par conséquent, **lorsque deux mots possèdent la même racine, mais des sens complètement différents, il vaut mieux les surreprésenter dans le fichier d'apprentissage** de façons à éviter des mauvais codages, pour éviter toute mauvaise orientation de l'algorithme d'apprentissage.

Un exemple : lors de la récente enquête PCV, des erreurs systématiques ont été commises sur un libellé d'un seul mot : "CHAUFFEUR". Dans le fichier d'apprentissage, le mot apparaissait très souvent, mais jamais seul : chauffeur poids lourds, chauffeur taxi, chauffeur bus, ... Dans le même fichier, on trouvait l'intitulé "CHAUFFAGISTE". Sicore a parcouru les deux premiers bigrammes des deux premiers mots : "CH", "AU" et deux bigrammes blancs (dans le deuxième mot), puis a sélectionné le troisième ("FF") : il ne restait qu'un seul libellé d'un mot commençant par "CHAUFF". Le contrôle de redondance, sur les trois premiers caractères des deux premiers mots, n'a pas posé de problèmes, et le chauffeur a donc été codé comme un chauffagiste.

On peut tirer une autre recommandation importante de cette expérience (et d'autres essais antérieurs) : **il est utile d'ajouter dans le fichier d'apprentissage des libellés courts** (2 mots ou 1 mot). Ceci pour au moins deux raisons : d'abord parce que cela oblige souvent l'algorithme d'apprentissage à aller chercher des bigrammes systématiquement utiles, comme le 4^{e} bigramme dans le cas du chauffeur et du chauffagiste, en accentuant la nécessité de départager à ce niveau. D'autre part, parce que les enquêtés ont souvent tendance à répondre de façon brève, en particulier lorsque le mot employé leur paraît suffisamment lourd de sens.

C'est un principe qui est déjà adopté depuis quelque temps pour construire le fichier d'apprentissage des PCS. Dans le FA des communes, il y a également beaucoup de libellés d'un mot pour des communes de taille importante.

1. Rappelons que l'expert modifie toujours le FAB, et jamais le FA : ce dernier est indirectement modifié, puisqu'il résulte d'une normalisation du FAB.

Remarque : de la même façon que dans la partie 3.2.2.1, on peut détecter automatiquement les mots posant ainsi problème en faisant un apprentissage sur deux mots, voire un, puis en recherchant les codages multiples.

3.2.2.3. Choix entre synonyme et FAB : ajout de synonymes

Dans une langue, on dit que deux mots sont synonymes s'ils appartiennent au vocabulaire (i.e. existent en tant que mots) et s'ils ont des sens équivalents.

Dans le cadre du codage automatique, le terme "synonyme" a une signification un peu plus large. En premier lieu, un mot n'est pas nécessairement un élément du vocabulaire : c'est une chaîne de caractères qui était isolée de la suivante (s'il y en a) et de la précédente (s'il y en a). La notion de synonyme concerne aussi bien les mots que les groupes de mots. Dire qu'un mot (ou groupe de mots) X a pour synonyme un mot (ou groupe de mots) Y signifie alors simplement que l'on souhaite que X soit, partout où on le trouve, remplacé par Y (ceci ne préjuge en rien d'un éventuel remplacement ultérieur de Y).

Il y a **plusieurs types de synonymes** : les *abréviations* (AUXILIAIRE = AUXIL, HAUT = HT), les *équivalents sémantiques* (DORMONS=DORS, DORMIR=DORS), les *sigles courants* (OUVRIER HAUTEMENT QUALIFIE = OHQ), et les *inclusions* dans une catégorie plus générale (VIOLON = MUSIQUE, CONCOMBRE = LEGUME).

Les abréviations sont intéressantes lorsque le mot (ou l'un de ses synonymes) est très courant, et n'apporte pas suffisamment d'information pour déterminer à lui seul le code. Ainsi, dans le cas de la profession, il est indispensable de rechercher tous les équivalents de mots comme "AGENT" (AGT, AG, ..), "ADJOINT" (ADJ, ADJT, ...), "OUVRIER" (OUVR, OUV, ...), "EMPLOYE" (EMP, EMPL, EMPLOY, ...), etc. Dans chaque cas, on choisira un mot de référence qui servira de synonyme à tous les autres. Ce mot devra être choisi de façon à ce qu'il puisse servir de racine à d'autres mots synonymes. Grâce à cela, il ne sera pas nécessaire de rajouter systématiquement le féminin (adjointe, ouvrière, employée), notamment si le mot de référence est suffisamment long.

On voit au passage qu'il existe deux objectifs contradictoires : avoir un libellé suffisamment court pour servir de racine, suffisamment long pour ne pas multiplier les synonymes (notamment les féminins). À cause de la décomposition en bigrammes, il est bon d'avoir un synonyme de référence ayant une longueur paire : au total, *dans le cas des synonymes de type "abréviation"*, un mot de référence de 4 ou 6 lettres est une bonne chose.

Les mêmes remarques valent pour les *équivalents sémantiques* (mots ayant la même signification, dans un contexte donné, mais ne correspondant pas nécessairement à des

abréviations) : utiliser systématiquement les synonymes lorsque les racines (4 lettres) sont différentes, bien choisir (taille, utilisation en tant que racine) le mot de référence.

Les **sigles courants** ne posent pas de difficulté en ce qui concerne le choix du mot de référence et sa longueur : ce sera de toutes façons le sigle. En revanche, **ils exigent de bien ordonner les synonymes**, et de tenir compte des synonymes précédents. Ainsi, on écrira "OUVRIER HAUTEMENT QUALIFIE" *après* les synonymes concernant le mot OUVRIER, et en veillant bien à ce que le mot de référence soit OUVRIER et non OUVR.

Enfin, les **inclusions** sont extrêmement efficaces pour prendre en compte correctement des mots ou groupes de mots ayant les deux caractéristiques suivantes :

- ils sont peu importants en soi ;

- ils appartiennent en quelque sorte à une "catégorie", et n'apportent pas plus d'information que celle-ci. Exemple : EUROPE 1 et FRANCE INTER appartiennent à la catégorie RADIO pour le codage de libellés d'occupation.

Si c'est le cas, on n'aura aucun intérêt à placer ces expressions dans le fichier d'apprentissage, ce qui conduirait à l'alourdir inutilement et à orienter la construction de l'arbre en fonction de critères peu pertinents. Toutes ces expressions seront donc gérées par des synonymes, le mot de référence étant la fameuse catégorie : RADIO dans l'exemple précédent, MUSIQUE pour VIOLON, ACCORDEON, HARPE, ...

De façon plus générale, l'usage des synonymes ne doit pas se faire sans précaution. En effet, **il arrive souvent que des mots aient plusieurs sens** : ainsi, on écrira naturellement BUVONS=BOIRE, BOIS=BOIRE, en omettant le fait que "BOIS" est non seulement un verbe conjugué, mais aussi un nom commun qui n'a rien à voir avec le verbe.

3.2.3. Validation

C'est l'étape finale de la mise au point, qui fait suite à une mise à jour des connaissances. Elle nécessite une double analyse : une analyse interne des bases (i.e. indépendamment de tout fichier à coder), et un test de codage automatique.

L'**analyse interne** n'est autre que le contrôle de cohérence : celui de chaque base de connaissances prise séparément (*cohérence intra-base*), et celui des couples de bases de connaissances (*cohérence inter-bases*). Ceci vaut pour toutes les bases de connaissances. Simplement, dans de nombreux cas, il y a indépendance entre bases : ainsi, il n'y a aucune cohérence à rechercher entre règles de normalisation et règles logiques.

Plutôt que de donner une liste exhaustive, donnons quelques exemples parmi les plus importants.

Un fichier d'apprentissage ne sera pas cohérent en soi s'il fait intervenir des codes qui ne sont pas valables, qui ne feront pas partie d'une liste officielle. Il sera ambigu si un même libellé peut mener à plusieurs codes distincts : ce défaut peut être facilement découvert en effectuant un autoquestionnement du fichier d'apprentissage.

En présence de variables annexes, le couple (FA, règles logiques) ne sera pas cohérent s'il existe des codes du FA qui ne sont ni des codes officiels, ni des noms de tables de décision. En revanche, le fait que le FA résulte d'une normalisation du FAB assure une parfaite cohérence entre FA et règles de normalisation.

On constate donc que *l'analyse interne permet de lever des ambiguïtés, de mettre en lumière des incohérences, mais elle ne règle nullement un problème important des bases de connaissances : leur incomplétude*, notamment celle du FA ; on peut en effet très bien avoir un ensemble parfaitement cohérent de connaissances, tandis que le codage automatique est parfaitement inefficace.

C'est le **test de codage automatique** qui permet de remédier à cela : on relance le chiffrement sur le fichier initialement utilisé (dont on avait tiré les non-codés et les mal-codés). Il peut alors arriver que des libellés préalablement codés ne le soient plus, car les règles de normalisation, par exemple, ont pu changer beaucoup de choses. Grâce à ce test, on va donc de nouveau affiner les bases de connaissances, en se calant sur nos objectifs d'efficacité et de fiabilité.

3.3.Organisation autour de Sicore

Cette organisation encore récente, dans la mesure où le produit a encore été peu utilisé en production.

À l'heure actuelle, il existe des **experts**, et des **bases de connaissances**, pour les variables suivantes : profession (et CS), commune (ainsi que départements, pays et nationalités), occupation (pour l'enquête Emploi du Temps), et SIRET (très spécifique au RP). D'autres bases ont également été mises en place, mais ne font pas l'objet d'un véritable suivi : les SICAV et les lieux de séjour.

Les deux variables jugées les plus importantes relativement au codage automatique sont la profession et la commune. Chacune fait l'objet d'un **groupe de travail** (respectivement *Sicore-Profession* et *Sicore-Commune*) qui se réunit deux à trois fois par an. Les groupes de travail constituent un forum permettant de faire remonter des problèmes

relatifs au codage automatique, et de mettre en évidence les spécificités du chiffrement dans telle ou telle enquête.

Ainsi, dans le cadre de Sicore-Profession, l'existence de deux champs sémantiques distincts (libellés provenant des ménages, ou des entreprises) a été mise en évidence et actée. Les réunions du groupe Sicore-Commune ont permis de voir que la variable annexe "date" était très importante pour le codage de la commune, en particulier pour l'état civil.

Le système Sicore s'organise autour d'un expert Sicore. Celui-ci est le **point d'entrée**, le relais, pour tout statisticien qui voudrait utiliser le codage automatique dans son application. Il centralise les bases de connaissances, gère le réseau des experts de variables et les groupes de travail correspondants, prend en compte les demandes sur le codage automatique et conseille les utilisateurs, organise les formations, spécifie les évolutions de l'outil.

L'expert Sicore est en contact avec l'**équipe informatique de Sicore**, chargée de la maintenance évolutive de l'outil, de l'intégration de Sicore dans les chaînes de traitement d'enquête, et, parallèlement, du "guide d'intégration" de Sicore.

Lorsqu'un codage automatique est effectué (dans quelque enquête que ce soit), l'expert Sicore récupère le fichier résultat et le communique à l'expert de variable, qui en effectue l'analyse (non-codés, mal-codés) et met à jour les bases de connaissances en conséquence.

Ce mode de fonctionnement, encore balbutiant, a fonctionné ainsi à l'issue de l'enquête PCV (enquête Permanente Conditions de Vie), dans laquelle la CS avait été codée automatiquement par Sicore : les libellés ont effectivement été étudiés par les experts Profession, qui ont réellement modifié leurs fichiers de connaissances après cette analyse.

4. Pratique de Sicore

Le vocable "Sicore" a plusieurs acceptions. Il représente le système dans son ensemble : programmes, bases de connaissances, méthodologie, organisation. Mais on peut l'envisager aussi sous le seul angle des programmes.

De ce point de vue, en quoi consiste Sicore ?

Sicore est un **ensemble de programmes qui sont des briques élémentaires pour le codage** : lecture de connaissances, apprentissage, normalisation, reconnaissance de libellé, traitement de variables annexes, ... Ces briques sont écrites en C, et fonctionnent aussi bien sur PC que sur site central.

Sur PC, toutes ces briques sont réunies au sein d'un seul et même logiciel, fonctionnant sous Windows. Sur gros ordinateur, il n'y a rien de plus que les briques élémentaires, qui sont appelables par des programmes.

4.1. Les modes d'utilisation de Sicore

Il y a donc, d'une certaine manière, **deux Sicore**.

Sicore est d'abord un **logiciel fonctionnant sur PC**. Lorsque nous parlons de **l'outil Sicore**, nous nous référons à ce logiciel micro, qui comporte des programmes de codage automatique, mais aussi une interface permettant de créer, d'analyser et de mettre à jour des connaissances. L'objectif majeur du logiciel Sicore sur micro est de mettre au point des connaissances ; le codage automatique fait partie des outils qui contribuent à cette mise au point, mais il n'est pas ici une fin an soi.

Dès que l'on veut mettre à jour des connaissances, ce logiciel est d'un **maniement délicat**, tout simplement parce que le fonctionnement du codage automatique est complexe, et qu'il se sert de bases de connaissances variées et difficiles à maîtriser. L'outil Sicore sur PC est employé par un petit nombre de personnes : les **experts** de variables (PCS, commune, SIRET, occupation) et l'expert Sicore, ce qui fait moins de dix personnes au total.

Si l'on veut juste coder, et si l'on a déjà des connaissances pour cela, il est en revanche **très simple d'emploi** : il suffit de charger le fichier d'environnement, puis de lancer le codage sur le fichier qui nous intéresse.

Mais le chiffrement automatique avec "Sicore" **fonctionne aussi sur site central.** Ce n'est pas le même Sicore. En effet, lorsqu'on code automatiquement, on n'utilise que les briques élémentaires : en gros, apprentissage, lecture de connaissances, normalisation, codage, transformations ; ce sont les mêmes programmes que sur micro, écrits en C. Il n'y a pas d'interface : l'interface Windows (sur PC) est spécifique, et n'a pas vocation à être portée sur site central. *L'utilisation de ce second Sicore est réservée à des* **informaticiens**.

En fonction du site et de l'objectif recherché, on trouve donc plusieurs types d'utilisateurs de Sicore.

Les utilisateurs experts se servent de l'outil Sicore sur micro. Ils enrichissent et modifient leurs bases de connaissances afin d'en améliorer la qualité (complétude, justesse, cohérence). Leur rôle consiste donc à *produire des fichiers de connaissances* (et plus précisément un "fichier d'environnement", les résumant tous), et non à réaliser des codages automatiques.

Les utilisateurs informaticiens emploient Sicore de façon très différente. Ils *produisent une chaîne de traitement d'enquête, en incorporant les programmes Sicore* (briques élémentaires de codage automatique) dans leur application. Ils récupèrent également le fruit du travail des experts, c'est-à-dire des fichiers d'environnement pour chaque variable. Contrairement aux experts, ils ne seront pas jugés sur la qualité des résultats de codage, mais sur le fait que leur chaîne fonctionne convenablement. Ce sont des utilisateurs du "second Sicore", le pur outil de codage. À la limite, ces informaticiens pourraient travailler aussi bien sur PC que sur gros système. Sur PC, ils n'utiliseraient qu'une petite partie des potentialités de l'outil Sicore.

Enfin, on peut à la rigueur imaginer qu'il y ait des **utilisateurs statisticiens**, ne jouant pas le rôle d'experts, et qui voudraient, eux, seulement *produire un codage* : ils se serviraient de l'outil micro. Il leur suffirait de charger le fichier d'environnement adéquat, de récupérer le(s) fichier(s) des réponses à l'enquête et de coder ce(s) dernier(s). Ce type d'utilisateur, ni informaticien ni expert, ne peut agir que si on lui mâche le travail : l'informaticien lui prépare le fichier à coder, l'expert de variable lui fournit le fichier d'environnement, l'équipe Sicore lui installe le logiciel et lui explique comment s'en servir (c'est très simple). Un tel mode de fonctionnement serait, par exemple, tout à fait adapté pour le codage de libellés de SICAV.

4.2. La mise en place du chiffrement automatique dans une enquête

L'utilisation de Sicore en production, dans une enquête donnée, sur une variable donnée, demande un minimum de préparation, et exige une bonne répartition des rôles entre les différents acteurs.

L'expérience montre que cinq acteurs importants sont susceptibles d'intervenir. Ce sont trois statisticiens et deux informaticiens, chacun pouvant être à la tête d'une équipe :

- **l'expert Sicore ;**
- **le statisticien responsable de l'enquête ;**
- **l'expert de la variable à coder ;**
- **le responsable informatique Sicore ;**
- **le responsable informatique de l'enquête.**

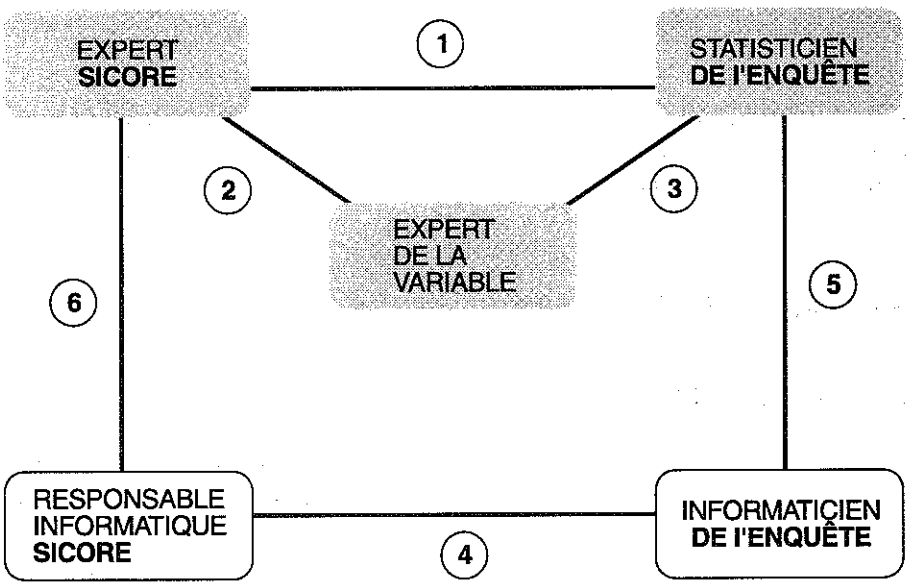
Parmi eux, deux sont "stables", au sens où ils ne dépendent pas de l'enquête ni de la variable considérée : ce sont, bien entendu, les deux représentants de Sicore.

Étant donné une variable à coder, l'expert de la variable, lorsqu'il existe, est toujours le même quelle que soit l'enquête. A l'inverse, le statisticien et l'informaticien de l'enquête seront en général liés à l'enquête qu'ils doivent traiter, mais souvent indépendants de la variable à coder : le même couple informaticien-statisticien s'occupera du codage de la CS et du codage du lieu de séjour, par exemple.

Entre ces 5 acteurs s'établissent des liaisons horizontales (entre statisticiens, ou bien entre informaticiens), et des liaisons verticales (entre représentants de Sicore ou entre représentants de l'enquête).

Figure 10 :

Acteur du système Sicore et liens entre acteurs



Dès lors, la mise en place du codage automatique dans l'application va suivre peu ou prou un certain scénario, dans lequel chacun a un rôle précis à jouer, et qui peut donner lieu à plusieurs adaptations : selon les cas de figure, tel ou tel acteur peut se révéler plus ou moins important, et tous ne sont pas nécessairement présents.

Dans un premier temps, on va *déterminer s'il est vraiment possible de coder automatiquement la variable* en question : en particulier, si la variable ne fait pas l'objet du moindre fichier d'apprentissage, il faudra créer les bases de connaissances, ce qui peut se révéler long. Le statisticien de l'enquête doit également évaluer le coût de la saisie,

ainsi que du logiciel (même fruste) de codage assisté qu'il faudra bien créer. Des impossibilités peuvent également émerger sur le plan informatique : incompatibilité entre sites, "gourmandise" de Sicore en mémoire, ...À ce stade-là, il est donc possible que l'on s'aperçoive que la mise en place du codage automatique est au total trop coûteuse en regard des avantages qu'elle procure.

Le statisticien de l'enquête cherchera ensuite (en collaboration avec l'expert Sicore et l'expert de la variable) à *déterminer quelles sont les spécificités de l'enquête* : liste de variables annexes utilisées, transcodages à définir, dessin de fichier. Cette opération est assez courte, mais se doit d'être menée avec soin.

Sur un plan informatique, le responsable informatique de Sicore et celui de l'enquête devront *mettre au point ensemble l'intégration des briques Sicore* dans les programmes de traitement spécifiques à l'enquête.

Les dernières étapes sont plus classiques : *tests* (de nature statistique, pour la qualité des résultats, et de nature informatique, pour le fonctionnement de la chaîne), et enfin *production*.

Lorsque le codage automatique dans son ensemble est terminé, les résultats de codage seront acheminés vers l'expert Sicore en vue d'*améliorer les bases de connaissances*, comme on l'a évoqué plus haut.

4.3. Applications de Sicore

4.3.1. Les variables testées

Sicore a été appliqué, à titre de test, au codage automatique de **nombreuses variables** : CS, PCS, communes, départements, lieux de séjour, SIRET, occupations, SICAV, types d'exploitation (en anglais). La plupart des tests, largement combinés à des mises à jour de connaissances, ont eu lieu sur PC.

Des **cas de figure très variés** ont été rencontrés : libellé homogène sans variables annexes (SICAV, département), libellé homogène + variables annexes (profession, occupation), libellé faiblement hétérogène (code département + libellé de commune), libellé fortement hétérogène (raison sociale-adresse, libellé de département + libellé de commune), création de bases de connaissances en partant de rien (occupations, SICAV), fusion de fichiers d'apprentissage (lieux de séjour), codage de libellés en plusieurs langues (types d'exploitation, à Statistique Canada).

En termes de **tailles de fichiers d'apprentissage**, la variété a également été grande : de quelques centaines (département), à quelques centaines de milliers (raison sociale - adresse), en passant par 12 500 (professions) et 42 000 (communes). Sur un PC 486 DX2, avec la version courante des paramètres d'apprentissage, le temps d'apprentissage est de l'ordre de 4 secondes pour des petits FA comme celui des SICAV (3 000 libellés), d'une trentaine de secondes pour les professions, et d'un peu plus de 2 minutes pour les communes.

Le **nombre de synonymes** a également beaucoup varié : de 0 (SICAV) à 1000 (professions), en passant par 500 (occupations) ; les communes n'utilisent quant à elles que quelques dizaines de mots vides ou synonymes.

Pour des variables importantes comme profession et commune, le chiffrement a été appliqué dans différents contextes : le RP, l'état civil, les DADS.

En fonction de ces contextes, les **efficacités de codage** furent donc très disparates.

Pour la profession, par exemple on obtient un taux de codage d'environ 66% des libellés d'un fichier provenant du RP 90, de l'ordre de 82,5%[1] pour la *CS* dans les DADS, entre 85% et 89% pour la CS de l'état civil, et enfin 76% dans l'enquête PCV (toujours la CS).

Selon les cas, l'efficacité de codage de la commune a varié de 93% à 99,5%.

Le taux de codage automatique de libellés d'occupation est quant à lui, à l'heure actuelle, de l'ordre de 65% (avec une fiabilité correcte), mais en ce domaine, les bases de connaissances (parties de zéro) évoluent beaucoup.

En ce qui concerne les SICAV, nous sommes passés de 61% à 80%, mais il n'existe qu'un seul fichier de libellés à coder, et le fichier d'apprentissage a été mis à jour en fonction des non-codés de ce fichier ; on peut donc augmenter l'efficacité autant que l'on veut, et le chiffre de 80% surévalue sans doute nettement la véritable efficacité.

4.3.2. Utilisation en production

Sicore a été appliqué à l'**enquête sur les transports de la DR de Rhône-Alpes**, appelée SYTRAL, où il fallait **coder des communes**, en l'occurrence des points de départ et d'arrivée pour chaque utilisation d'un transport en commun. Sur quelques milliers de libellés, seuls trois n'ont pas été codés par Sicore : le chiffrement assisté n'a donc pas été nécessaire.

1. On peut obtenir 84,8% en changeant les paramètres d'apprentissage, mais la qualité est moindre. On obtient 80,0% sans aucun bigramme prioritaire (et la qualité n'est pas particulièrement excellente).

Il est difficile d'en tirer des conséquences sur un plan statistique, car le chiffrement de la commune n'est certes pas le plus difficile. En revanche, informatiquement, on a pu voir que dans ce cas (où il n'y avait pas véritablement d'intégration dans une chaîne), le temps de travail nécessaire était très minime (un à deux jours)

Sicore a également servi à **chiffrer la CS et les lieux de séjour dans l'enquête PCV** (de la division Conditions de Vie des Ménages). La situation était idéale pour une évaluation de Sicore, puisqu'un codage manuel avait été fait en parallèle.

L'utilisation de Sicore était ici intéressante à double titre. D'une part, elle permettait de tester l'outil et en particulier son intégration : informatiquement, est-ce-que "ça marche" ? En second lieu, elle conduisait à évaluer la qualité du codage automatique. Nous présentons ici les résultats de cette expérience.

Du point de vue informatique, grâce à une très bonne collaboration entre l'équipe PCV du CNI d'Aix et l'équipe Sicore (au CNI de Paris), il n'y a eu *aucune difficulté particulière*. Restait alors à analyser les résultats.

La codification des CS a porté sur environ 9992 libellés, celle des lieux de séjour sur 12239. **Sicore a codé automatiquement 76% des CS et 92% des lieux de séjour**, ce qui a été jugé tout à fait satisfaisant, en termes d'efficacité, par le comité de pilotage de PCV. Mais ce n'est pas là le seul critère : il s'agissait de savoir si les codes Sicore étaient "bons". Pour évaluer cela, une technique s'imposait : rechercher dans combien de cas le code automatique différait du code manuel, puis analyser une par une ces divergences.

Les lieux de séjour n'ont pas posé de réels problèmes : nous avons déjà vu qu'un grand nombre d'entre eux étaient codés par Sicore. Lorsqu'on étudie les écarts entre codage automatique et codage manuel, on s'aperçoit qu'il y a très peu de différences : seulement 3% des libellés que Sicore a réussi à coder ont donné des résultats distincts du codage manuel, c'est-à-dire 315 lieux de séjour exactement. On peut alors se poser une question : certes, il y a peu d'écart, mais lorsque les codes sont différents, qui a raison ? En étudiant un par un les 315 libellés en question, on s'aperçoit que c'est le codage automatique qui gagne : en effet, **le code Sicore est juste dans 82% des cas, et le code manuel dans les cas restants**.

Parmi les erreurs de Sicore, citons l'exemple de "CANARIES" (qui n'était pas dans le fichier d'apprentissage), codé comme la ville corse "CANARI".

Le codage d'une CS est en revanche une opération plus délicate, ce qui implique qu'il y ait beaucoup plus de divergences entre codage automatique et codage à la main : parmi les intitulés de profession codés par Sicore, plus de 1 sur 6 (18%) divergent du code manuel. La question se pose à nouveau : qui a raison ? Les différences de chiffrement de CS ont donc été analysées : les experts Profession s'en sont chargés.

L'étude des divergences montre une nouvelle fois que c'est plutôt le codage automatique qui donne les meilleurs résultats : *dans 62,5% des cas, le code Sicore est le bon code.* Les erreurs manuelles ont plusieurs origines, l'une d'entre elles étant une prise en compte erronée, dans certains cas, du clivage public-privé : par exemple, les cuisiniers des collectivités locales sont classés en employés de la Fonction publique (52) au lieu de cuisiniers qualifiés (63). À l'inverse, *le codage manuel a raison dans 17,5% des cas.* Enfin, *dans 20% des cas, les deux codes peuvent convenir* : il y a ambiguïté ; celle-ci est souvent due au fait que Sicore n'utilise pas le libellé d'activité en clair.

4.3.3. Présent et avenir de Sicore

À l'heure actuelle, répétons-le, Sicore se présente d'abord comme un logiciel micro, fonctionnant sous Windows, et utilisé par un très petit nombre de personnes : les experts. **Il n'est pas du tout envisagé que cet outil de mise au point des connaissances soit largement utilisé, car son usage est complexe**, et nécessite une très bonne maîtrise de Sicore et l'expertise sur une variable. Pour cela, une formation de 3 jours a été dispensée[1], et trois documentations sont fournies : le guide utilisateur de l'outil micro, un glossaire détaillé, et la documentation méthodologique.

En revanche, le "second Sicore", i.e. le pur outil de codage automatique (sur site central), est assez facilement intégrable à une application : pour le moment, l'équipe informatique de Sicore participe largement à ces intégrations, mais à terme, les informaticiens d'enquête devront pouvoir le faire seuls. Les documentations associées (documentation du programmeur, guide d'intégration) ne seront disponibles que début 1996.

Notons également que du point de vue des fonctionnalités, tout n'est pas terminé : les aspects "contrôle de cohérence" et "transformations" sont en voie d'achèvement.

Dans l'avenir, Sicore sera appliqué au codage de la CS dans les DADS (il remplacera ainsi Quid), et utilisé à nouveau pour l'enquête PCV. Suivront sans doute d'autres applications, par exemple à d'autres enquêtes du tronc commun. L'utilisation de Sicore pour l'enquête Emploi interviendra à plus long terme.

Enfin, on envisage de *commercialiser* Sicore : plusieurs organismes en France (par exemple l'ITSEE, en Nouvelle-Calédonie) et quelques services statistiques à l'étranger (aux États-Unis en particulier) ont manifesté leur intérêt à ce sujet.

23. Des personnes qui connaissaient déjà bien Sicore, précisons-le.

SIGLES UTILISÉS

CNI : Centre National Informatique
CS : Catégorie Socio-professionnelle
DADS : Déclarations Annuelles de Données Sociales
FA : Fichier d'Apprentissage
FAB : Fichier d'Apprentissage Brut
MCA : Mise en Concordance Automatique = logiciel qui permet, entre autres, de retrouver les codes SIRET d'établissements à partir de leur raison sociale et de leur adresse
PC : Personal Computer = ordinateur personnel
PCS : Profession et Catégorie Socio-professionnelle
PCV : enquête Permanente Conditions de Vie
QUID : QUestionnaire d'IDentification ; désigne un algorithme général de codage automatique, ainsi qu'un logiciel, utilisé à l'Insee, par exemple pour le codage de la profession dans l'enquête Emploi.
RP : Recensement de la Population
SICORE : Système Informatique de COdage des Réponses aux Enquêtes
SIRET : c'est l'identifiant d'un établissement
SYTRAL : enquête sur les transports dans la DR Rhône-Alpes
TFD : Tri par Fréquence Décroissante

BIBLIOGRAPHIE

LORIGNY, J. (1988). Quid, une méthode générale de chiffrement automatique. *Survey methodology,* décembre 1988, Vol.14, n°2, pp.289-298.

LYBERG L., DEAN P. (1992) Automated coding of survey responses : an international review, Working Paper, *Conférence des Statisticiens Européens, Work Session on Data Editing,* Washington, Mars 1992

RIVIERE, P. (1994). Le système de codification automatique Sicore.Working Paper, *Conférence des Statisticiens Européens,* Séminaire ISIS 94, Bratislava, Mai 1994

WENZLOWSKI, M.J. (1988) ACTR - A generalized automatic coding system, *Survey Methodology,* décembre 1988, Vol.14, n° 2, pp. 299-308.